VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2017 Radek Čep

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Radek Čep**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: AstrumQ Interactive, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Tomáš Fabián, Ph.D.**

Konzultant bakalářské práce: Bc. Aleš Vyka

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2017 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 24. dubna 2017

Děkuji Alešovi Vykovi a Tomášovi Rozehnalovi za příležitost a Jiřímu Urbáškovi za spoustu cenných rad.

Abstrakt

Tato bakalářská práce bude popisovat technologie a postupy, které jsem se naučil používat během své odborné praxe ve firmě AstrumQ. Hlavní náplní práce je mapovat vývoj aplikace Sportuj v Ostravě, během kterého jsem tyto vědomosti využil.

Klíčová slova: iOS, Swift, Cocoa Touch, UIKit, Bakalářská praxe

Abstract

This bachelor thesis is going to describe technology and processes which I learned during my professional practice in AstrumQ. Eventually it is going to map development of an application Sportuj v Ostravě where I have applied some of the newly acquired knowledge.

Key Words: iOS, Swift, Cocoa Touch, UIKit, Bachelor practice

Obsah

Seznam použitých zkratek a symbolů

MVC	–	Model View Controller
Obj C	–	Objective C
UDID	–	Unique Device Identifier
UI	–	User Interface
UX	–	User Experience
VC	–	UIViewController

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Pro absolvování individuální odborné praxe jsem se rozhodl z několika důvodů. Jednak jsem chtěl získat praktické zkušenosti z oboru, nabít užitečné vědomosti z témat kterými bych se chtěl v budoucnu zaobírat a v neposlední řadě navýšit vlastní cenu na trhu práce.

Obor, ve kterém jsem chtěl během praxe působit, vývoj mobilních aplikací pro iOS, jsem si vybral z vlastní dlouholeté a nesporně kladné zkušenosti s produkty firmy Apple. V této práci budu popisovat vlastní působení ve firmě AstrumQ na pozici Vývojář mobilních aplikací – junior pro iOS, projekty, na kterých jsem pracoval a technologie, které k nim byly použity.

1.1 O firmě AstrumQ

Společnost AstrumQ, založená roku 2012 sídlí v Ostravě a zaměřujeme se na vývoj mobilních a webových aplikací zejména v oblastech obchodu, vzdělávání, gastronomie, zábavy a marketingu. Dále nabízí servis mobilních a webových aplikací, nebo odborné poradenství. Její vizí je stát se přední českou vývojářskou firmou s úspěšnými start-up projekty jejich zákazníků na mezinárodním poli v oboru vývoje mobilních aplikací [1].

Firmu jsem našel na základě propagačních materiálů v budově fakulty. Během úvodního pohovoru jsem zjistil že nabízejí praxi se zaměřením, jaké jsem si představoval a také to že již mají předchozí zkušenosti se studenty na bakalářské praxi.

1.2 Má práce ve firmě

Mým úkolem ve firmě bude práce spolu se zkušeným programátorem na mobilní aplikaci Sportuj v Ostravě, která bude sloužit uživatelům k nalezení sportovních akcí ve městě, procházení zdejších sportovišť a sportovních klubů, které zde působí.

Úspěšnost práce bude hodnocena podle množství vlastních nabytých schopností a zkušeností. Podle osobního přínosu, který mi tato zkušenost dá, nebo naopak nedá.

2 Úvod do technologie

2.1 iOS a iPhone

iOS je moderní mobilní operační systém představený roku 2007 spolu s prvním chytrým telefonem iPhone. Pro vývojáře se systém otevřel až o rok později s příchodem jeho druhé verze a obchodu App Store. Oproti konkurenčním platformám je iOS poměrně uzavřený, silně omezuje, jaké mohou uživatelé instalovat aplikace a pro většinu uživatelů tím zůstává jediný zdroj software právě oficiální App Store. Aplikace procházejí přísným schvalovacím procesem a vývojáři musí dodržovat stanovená pravidla, používat pouze povolená API a tvořit aplikace v souladu se vzhledem a chováním systému. Tato na jednu stranu přísná pravidla poskytují koncovým uživatelům pohodlí a bezpečí při používání jejich zařízení.

2.2 Porovnání s konkurenční platformou

iOS je podle analytické společnosti IDC aktuálně nainstalován na 11,7% všech mobilních zařízení. V lednu roku 2016 společnost Apple oznámila, že její operační systém je nainstalován na 1 miliardě aktivních zařízení. Na rozdíl od konkurence má iOS vysokou míru zařízení, na kterých běží aktuální verze systému. Fragmentace verzí je zde mnohem nižší, 60% uživatelů má nainstalovanou poslední verzi systému a 92% zařízení běží na systému mladším dvou let. U konkurenčního OS Android má poslední verzi systému na svém telefonu či tabletu pouze 24% uživatelů, přes 25% přitom stále používá verzi představenou v roce 2013 [2-5].

Tabulka 1: Fragmentace iOS.

Verze OS (rok vydání)	Zastoupení [%]
iOS 10 (2016)	60
iOS 9 (2015)	32
dřívější	8

Tabulka 2: Fragmentace OS Android.

Verze OS (rok vydání)	Zastoupení [%]
Lollipop (2014)	34,1
KitKat (2013)	25,2
Marshmallow (2015)	24
Jelly Bean (2012)	13,7
Gingerbread (2010)	1,3
Ice Cream Sandwich (2011)	1,3
Nougat (2016)	0,3
Froyo (2010)	0,1

2.3 Vývoj aplikací pro iOS

Apple nabízí pro vývoj nativních aplikací pro svou mobilní platformu dva programovací jazyky, starší Objective C, který existuje již bezmála 33 let a mnohem mladší, moderní programovací jazyk Swift, který byl poprvé představen v červnu 2014, ale i tak se již dokázal vyšplhat v TIOBE indexu popularity na 12. místo. Oba se řadí mezi objektově orientované, staticky typované a kompilované jazyky. Přičemž programátor není při vývoji omezen k výběru pouze jednoho, projekt může obsahovat části napsané každá v jiném jazyce [6].

Vývojové prostředí dodávané Apple, Xcode, se poprvé objevilo v roce 2003 pro Mac OS X Panther. Dnes již existuje ve své 8. verzi a k jeho spuštění je potřeba macOS Sierra 10.12 nebo OS X El Capitan 10.11.5 a vyšší. To znamená, že k plnohodnotnému vývoji aplikací pro iOS je třeba vlastnit počítač vyrobený Apple. V dnešní době sice existují různé způsoby, které dokáží určitým způsobem zpřístupnit vývoj i mimo jablečný ekosystém, jako například instalace OSx86, neoficiální distribuce macOS určené pro počítače, které nebyli vyrobeny Apple, nebo portál macincloud.com který zprostředkovává systém skrze webovou aplikaci. Dále existují různé frameworky slibující multiplatformní aplikace, mezi ty známější patří například Xamarin, ty jsou ale mimo rozsah této práce.

2.4 Obj C

Objective C se objevil na začátku osmdesátých let minulého století a na jejich konci byl vybrán jako hlavní programovací jazyk platformy NeXTSTEP firmy NeXT, ze které vycházejí dnešní macOS i iOS.

V projektu se obvykle nacházejí dva typy souborů .h – pro hlavičkové a interface soubory a .mm pro vlastní kód. Obj C je nadmnožina jazyka C, což znamená, že je možné v projektu volně používat kód napsaný v C.

Přístup k objektům je inspirován jazykem Smalltalk, v Obj C se nevolají metody objektu, nýbrž se objektu posílají zprávy. Na rozdíl od jiných kompilovaných jazyků se v Obj C objekt přijímající zprávu volí za běhu programu, není tak nutné, aby objekt v době kompilace implementoval požadovanou metodu, program v Obj C může za běhu nahrávat nové třídy, nebo přidávat metody ke stávajícím.

2.5 Swift

Swift je nový, rychle se vyvíjející programovací jazyk, který k roku 2017 existuje již ve třetí iteraci. V roce 2014 byl představen jako open-source alternativa k Objective C, coby primárního jazyku k vývoji iOS a macOS aplikací. Z Obj C přejímá Swift mnoho významných rysů a přidává prvky, které zvyšují bezpečnost a stabilitu aplikace.

Mezi klíčové vlastnosti patří unifikovaný způsob ukazatelů na funkci, každá funkce i anonymní (closure) má ve Swift datový typ, skládající se z jejich parametrů a návratového typu, lze tak s nimi manipulovat jako s libovolnou proměnou. Dále jazyk podporuje tvorbu entic (tuple), která

poskytuje možnost vracet z funkce více než jednu hodnotu nebo odvozování datových typů, možnost vynechávání závorek a další pro vývojáře příjemné funkce [7].

2.6 Swift a Obj C

To, co oba jazyky sdílejí je unikátní systém pojmenovávání funkcí a proměnných, konvence obou vyžadují, aby byla jména funkcí i proměnných samo popisující, jejich použití pak téměř tvoří věty usnadňující orientaci a čitelnost kódu.

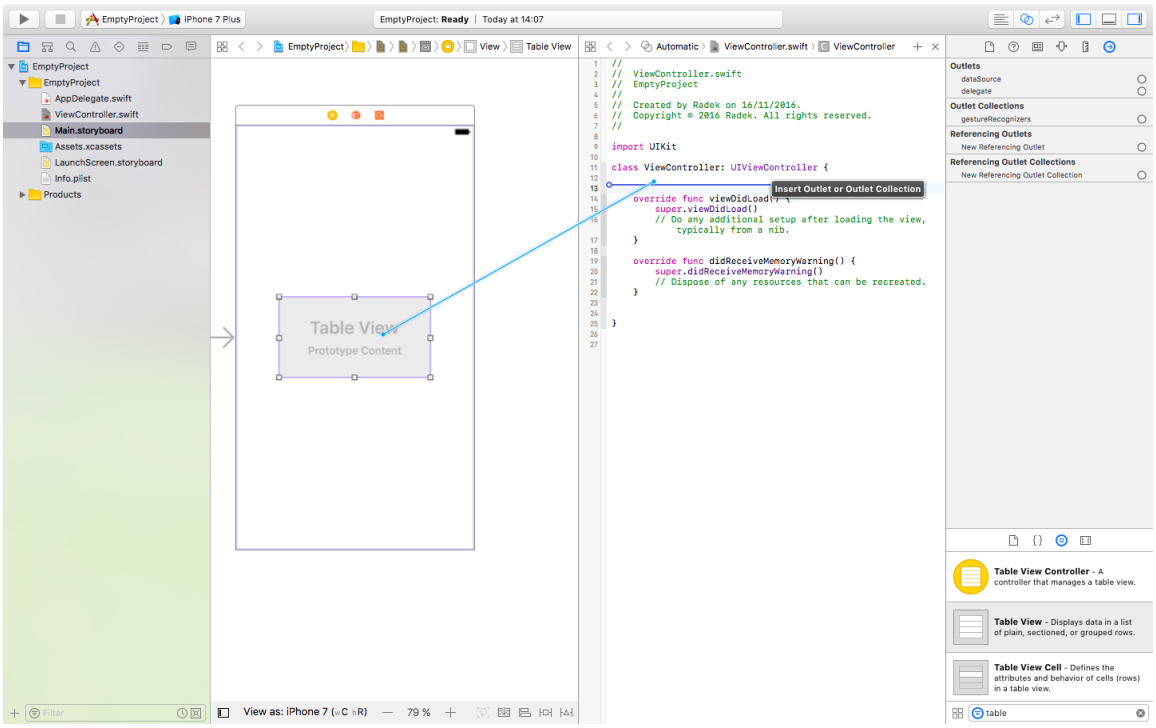
Použití Swift kódu v kódu Obj C a naopak je jednoduché a bylo jednou s klíčových vlastností při návrhu Swift. Tato funkcionalita se nazývá Mix and Match. Pro její použití je nutné, v případě použití Obj C kódu ve Swift, v projektu vytvořit takzvaný bridging header jedná se o .h hlavičkový soubor, ve kterém se nacházejí importy všech Obj C tříd které se tím zpřístupní ve všech .swift souborech projektu. V opačném případě se třídy napsané ve Swift zpřístupní v Obj C pomocí klausule #import "<Jméno aplikace>-Swift.h".

```
//Swift:
var cell = tableView.dequeueReusableCell(withIdentifier: "cell",for: indexPath)
//Obj C:
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier: "cell"];
```

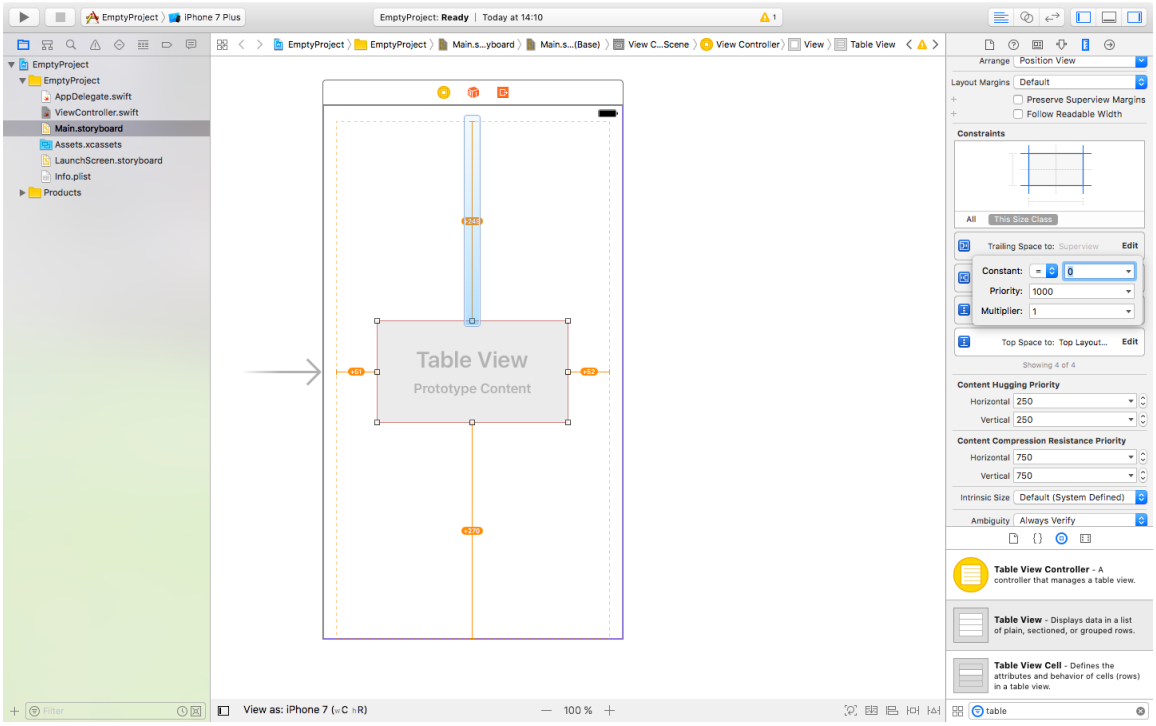
Výpis 1: Ukázka výřečnosti kódu Obj C a Swift - získání buňky z UITableView.

2.7 Interface builder

Interface builder je jednou z největších předností vývoje iOS aplikací. Je to jednoduchý, ale velice sofistikovaný nástroj k vytváření vzhledu aplikací. Prvky jsou vkládány jednoduše pomocí drag & drop, jejich vzájemnou polohu v prostoru určují takzvané constraints ty mohou být pevná konstanta, vzdálenost od určitého jiného prvku, ale i otevřené množiny typu "větší/menší než", jsou prioritizovatelné a jejich změna je animovatelná. Jednotlivé prvky můžou mít nastavenou horizontální i vertikální odolnost proti stlačení i roztažení, nebo mohou být seskupené ve svazcích. Jednotlivé prvky UI vybraného VC se pak spojí s kódem jednoduše, přetažením elementu do souboru, v následném dialogu lze vybrat, jestli se má vytvořit odkaz, funkce spojená s akcí, dokonce i jaké parametry mají být při akci poslány do funkce.



Obrázek 1: Ukázka spojení rozhraní s kódem.



Obrázek 2: Ukázka úpravy constraint.

2.8 Návrhové vzory při vývoji iOS aplikací

Základní jednotkou každé iOS aplikace je UIViewController (VC). Apple tak vývojáře nabádá k použití MVC návrhového vzoru. Objekt UIViewController je nejjednodušší funkční spustitelnou aplikací, obsahuje klíčové funkce, které jsou na něm volány v jeho různých životních fázích (viewDidLoad, viewWillAppear, viewWillDisappear, atp.), zároveň jak vyplývá z jeho názvu kontroluje s ním spojené view, což je instance UIView, objektu, v rámci kterého je vykreslován obsah. Vytvořením třídy, která dědí z UIViewController a implementací daných metod vzniká vlastní aplikace. Krom vlastních instancí tříd dědicích z UIViewController je v UIKit frameworku k dispozici mnoho konkrétních implementací sloužících k různým účelům.

Přechody mezi jednotlivými VC jsou řešeny pomocí přechodů (segue), které se dále dělí na více druhů, nejjednodušeji je můžeme rozdělit na takové, které kompletně překryjí stávající VC, nenabízí tedy uživateli systémovou možnost návratu, ta musí/může být neimplementována programátorem, nebo takové, které nový VC můžou přidat do navigačního zásobníku UINavigationController ve kterém je vložený VC volající segue. Přechody mohou mít nastavené identifikátory. Předávání informací mezi jednotlivými přechody je řešeno metodou prepareForSegue ve které dostane VC volající segue k dispozici instanci cílového VC. Při opouštění zobrazovaného VC je pak možné využít takzvané unwind segue což je metoda cílového VC ve které obdobně obdrží instanci volajícího VC před dokončením přechodu.

Další návrhový vzor, který je při vývoji iOS aplikací hojně využíván je delegation. Delegovaný objekt si udržuje referenci na objekt svého delegáta, kterému pak zasílá zprávy o událostech, které delegovaný objekt udělal, nebo se chystá udělat. Delegát tak může na událost reagovat, upravit svá data, vzhled svých view, nebo naopak poskytnout delegovanému objektu potřebné informace. Výhodou je, že jeden objekt může delegovat více jiných různých objektů v rámci aplikace.

Příkladem může být objekt UITableView (tableView) při vývoji často využívaný prostředek k zobrazení seznamu položek. Vložíme-li jej do VC a označíme VC jako jeho delegát, bude tableView po VC vyžadovat informace jako počet položek k zobrazení, vzhled jednotlivých buněk apod. VC tak může držet všechny relevantní informace a postačí si s výchozí implementací UITableView.

2.9 Distribuce aplikací

Pro distribuci aplikací jsou klíčové dva internetové portály, developer.apple.com, který slouží ke správě App IDs a itunesconnect.apple.com, přes který se podepsaná aplikace nahrává do AppStore.

2.9.1 App ID

Jedná se o řetězec složený ze dvou částí, rozdělených tečkou, identifikující jednu, nebo více aplikací od jednoho vývojáře. První část řetězce je takzvané Team ID, unikátní identifikační kód přiřazený vývojáři, druhou částí je Bundle ID, to si určuje vývojář a může být jedinečné pro jednu výhradní aplikaci (Explicit) nebo pro skupinu aplikací jednoho vývojáře (Wildcard). Wildcard ulehčuje vývojáři práci při vytváření stále nových App ID pro každou aplikaci, ale jelikož se jedná o unikátní identifikátor aplikace, jeho nahrazení Wildcard zabraňuje implementaci některým funkcím specifickým pro jednu aplikaci, jako Game Center, iCloud, in-app purchases, nebo push notifikace.

2.9.2 Provisioning Profiles

Vývojářský portál Apple nabízí několik možných profilů, které si zde vývojář může vygenerovat k podepisování aplikací.

Development Slouží k instalaci aplikace do jednoho z testovacích zařízení pomocí Xcode. Vývojář může označit maximálně 100 zařízení (identifikovaných pomocí UDID) a používat je k testování aplikací.

Ad Hoc Podobně jako Development profile slouží k podepsání aplikace pro malé množství testovacích zařízení. Nevyžaduje ale instalaci skrze Xcode, je možné instalovat i skrze iTunes, nebo Safari. Aplikaci nelze svázat s debuggerem. Využívá se pro distribuci veřejných beta verzí.

AppStore Slouží k podepsání aplikace před nahráním do App Store.

Xcode 8 umožňuje automatickou správu profilů, čímž značně usnadňuje práci s podepisováním aplikací a eliminuje potřebu použití wildcard v rámci App ID.

2.9.3 Odeslání aplikace do AppStore

Podepsanou aplikaci je možné skrze Xcode odeslat na server iTunesConnect kde je dále nutné vyplnit všechny požadované informace, zvolit podporovaná zařízení, popis, screenshoty a ikony v patřičných velikostech atp. Následně je možné odeslat aplikaci k review, před vpuštěním do AppStore je každá aplikace kontrolována, zda jsou spolehlivé, fungují, jak je předpokládáno a splňují všeobecné podmínky obchodu. Kontrola aplikací by měla ve většina případů proběhnout do 48 hodin od nahrání.

3 Práce ve firmě

Krom projektů, na kterých jsem se ve firmě podílel jsem přišel také do styku s obecnými činnostmi a aplikacemi spojenými s každodenním fungováním firmy.

3.1 Nástroje používané ve firmě

ActiveCollab Každý úkol, který byl zvolen během týdenní porady byl také zadán do firemního portálu ActiveCollab, zde se určoval sprint, do kterého patří, kdo na něm bude pracovat, odhadovaná a později i reálná délka úkolu a bylo možné zde k úkolům průběžně psát poznámky, nebo komentáře.

Giriton Krom zaznamenávání jednotlivých odpracovaných úkolů se čas strávený v kanceláři zaznamenával klasickým "odpíchnutím" při příchodu a odchodu do elektronického docházkového systému.

Slack Byl hlavním komunikačním kanálem ve firmě, jedná se o propracovaný komunikační nástroj pro firmy, který umožňuje jak jednoduchý, tak skupinový chat, video hovory, sdílení souborů, propracované vyhledávání atp. Jeho aplikace je dostupná na všech hlavních platformách jak na desktopu, tak mobilních zařízeních.

Git Samozřejmostí pro práci na jedné aplikaci ve více lidech bylo použití Gitu. Konkrétně byli repositáře umístěné na serveru GitBucket a k samotnému používání sloužila aplikace SourceTree, která umožňuje ovládat všechny podstatné funkce Gitu.

Apiary Webové rozhraní sloužící k dokumentaci API. Krom popisu celé komunikace se serverem nabízí také možnost vygenerovat si vlastní mock server pro testovací účely. Nabízí přehledné příklady requestů i konzoly pro jejich vlastní tvorbu a úpravu spolu s možností odeslání a zobrazení odpovědi.

Zeplin Byl nástroj používaný ke zprostředkování grafiky grafikem vývojářům. Nabízí mnoho pokročilých a užitečných funkcí, zobrazuje mockupy s možností zobrazení jednotlivých pixelových vzdáleností, nebo rychlou volbu ke stažení používaných ikon ve formě xcassets, tedy v sadě všech velikostí, které by iOS mohlo použít, jako jeden předpřipravený balíček.

OwnCloud / Google Drive Online diskové řešení pro sdílení dokument. V případě Google Drive i pro jejich vytváření a editaci.

3.2 Organizace práce

Na aplikaci se pracovalo v týdenních cyklech (sprintech) kdy se vždy domluvila práce, která se měla během daného sprintu stihnout. Určovala se její obtížnost a popřípadě se řešilo proč se některé věci v minulém týdnu nestihly, nebo naopak stihly mnohem rychleji. Různé úkoly měli různé priority, celkové fungování aplikace tak bylo například důležitější než některé grafické nedostatky.

Podstatnou součástí organizace byla také "lístečková stěna", krom spracování úkolů v systému ActiveCollab byla jedna stěna v kanceláři firmy pokryta obrovskou tabulkou, kde řádky tvořili úkoly a tři sloupce označovali stav úkolu ("To do", "In progess"a "Done"). Každý úkol se rozdělil na nejmenší části, které se napsali na lístečky a ty se postupně přesouvali z prvního do posledního sloupce v řádku tak, jak se postupně plnili.

4 Realizace aplikace Sportuj v Ostravě

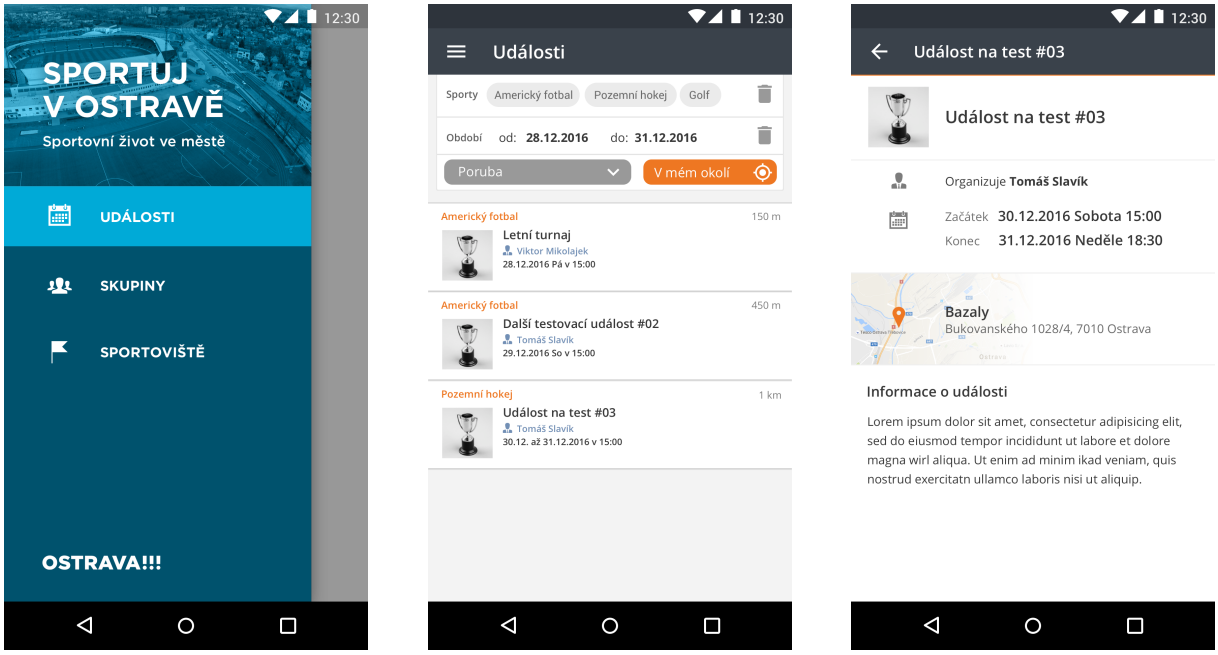
Aplikace Sportuj v Ostravě je cílená na zařízení s iOS 10 a vyšší, tato volba byla učiněna z důvodu možnosti využít nové možnosti, které přibyli s poslední verzí systému, jako například pestřejší notifikace. Uživatelská základna se staršími verzemi systému je navíc již minoritní a není tedy důvod nevyžadovat poslední verzi OS. Prvky uživatelského rozhraní budou zapadat do celkového vzezření systému, v maximální možné míře budou využité prostředky frameworku UIKit. Celá aplikace by tak měla zapadat do systémového UX a splňovat iOS Human Interface Guidelines.

4.1 První fáze aplikace

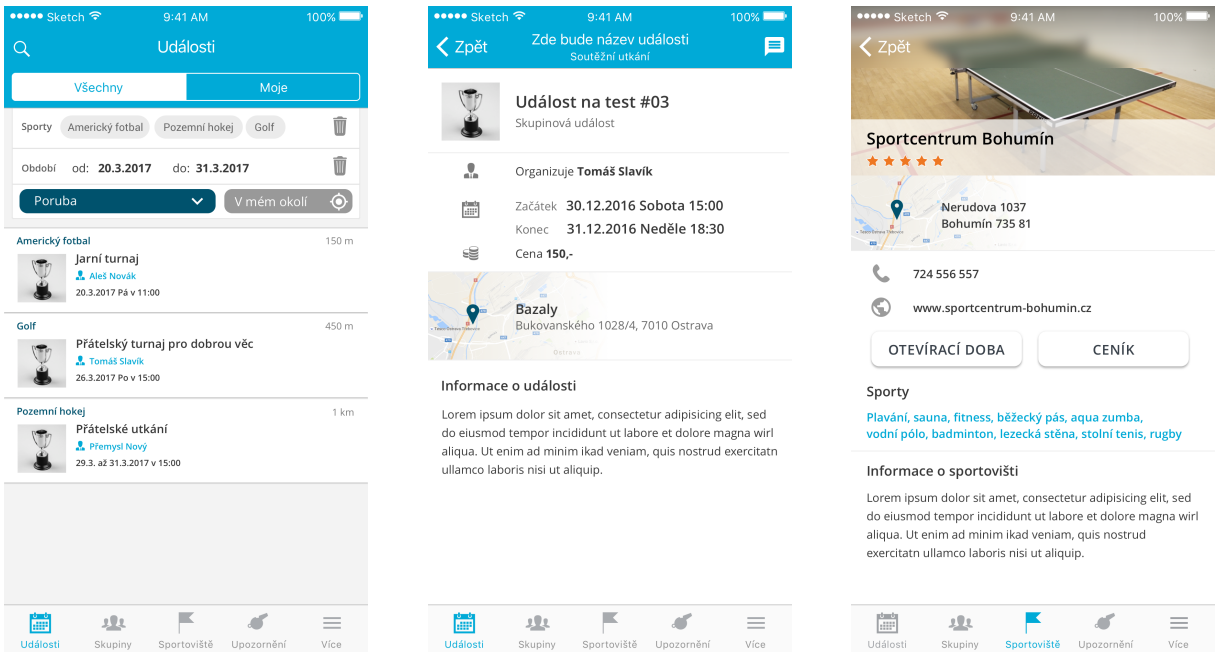
V první verzi aplikace jde o vytvoření prohlížeče pro veřejně přístupné informace v rámci systému. Cílem je ukázat v aplikaci sportovní akce v okolí s možností jejich filtrování podle data a polohy. A nabídnout registrovaným uživatelům možnost přihlášení.

4.2 Konverze zadání

Vývoj aplikace probíhá souběžně pro iOS, tak pro OS Android. První mockupy díla byly pro druhý zmiňovaný. V rámci příprav na tvorbu v nich bylo učiněno několik významných změn. Největší bylo odstranění takzvaného Hamburger menu (viz obr. 3 a 4). A to hned z několika důvodů, prvním z nich je že iOS používá gesto tažení z levé strany obrazovky pro přechod zpět v rámci navigace aplikací, dále je prostor v levém horním rohu obrazovky obsazen navigačními tlačítky. Apple v aplikacích sám nedoporučuje používání Hamburgeru místo něj je doporučován Tab Bar Controller, na který byla také aplikace předělána. Dalšími designovými změnami v rámci zachování systémového chování a vzhledu byli drobné úpravy tlačítek, roletek a navigačního panelu [8].



Obrázek 3: Původní mockupy pro OS Android.



Obrázek 4: Nové mockupy pro iOS. Odstraněné hamburger menu a boční lišta nahrazena tabbarem na spodní části obrazovky.

4.3 Instalace frameworků

K přidávání frameworků třetích stran do Xcode projektu se využívá některého z dependency managerů. Nejčastější volbou jsou CocoaPods, nebo Carthage.

4.3.1 CocoaPods

CocoaPods jsou centralizovaný systém systém pro správu frameworků, to znamená, že si udržuje vlastní seznam knihoven, které je schopen poskytnut. Přidání probíhá vytvořením speciálního souboru Podfile v kořenovém adresáři projektu. V něm se specifikuje verze cílového systému projektu, všechny knihovny a jejich verze. CocoaPods jsou ovládány skrze příkazovou řádku, po zadání $ pod install se všechny frameworky stáhnou a vytvoří se soubor s příponou .xworkspace, ten již obsahuje veškerou potřebnou konfiguraci spojenou s přidáváním frameworků.

4.3.2 Carthage

Na rozdíl od CocoaPods je Carthage decentralizovaný systém. Prvotní konfigurace probíhá obdobně vytvořením Cartfile souboru, ve kterém jsou ale oproti Podfile přímo zdrojové URL instalovaných frameworků. Carthage poté knihovny stáhne a sestaví. Nevytváří žádné další soubory a nijak nemodifikuje Xcode projekt. Přidání frameworku je tak relativně složitější, ale plně pod kontrolou vývojáře.

4.4 Připojení k API

Pro připojení k API serveru byl zvolen framework Alamofire. Je to oblíbený opensource framework pro zjednodušení práce s http v rámci Swift. Na GitHub má téměř 22000 hvězdiček. Krom celkového zjednodušení práce s web requesty poskytuje mnoho pokročilých funkcí jako zřetězení požadavků, pokročilé zpracování parametrů v URL, nebo JSON, upload i download souborů, nebo podpora autentizace. Práce s requesty pak probíhá pomocí closures [9].

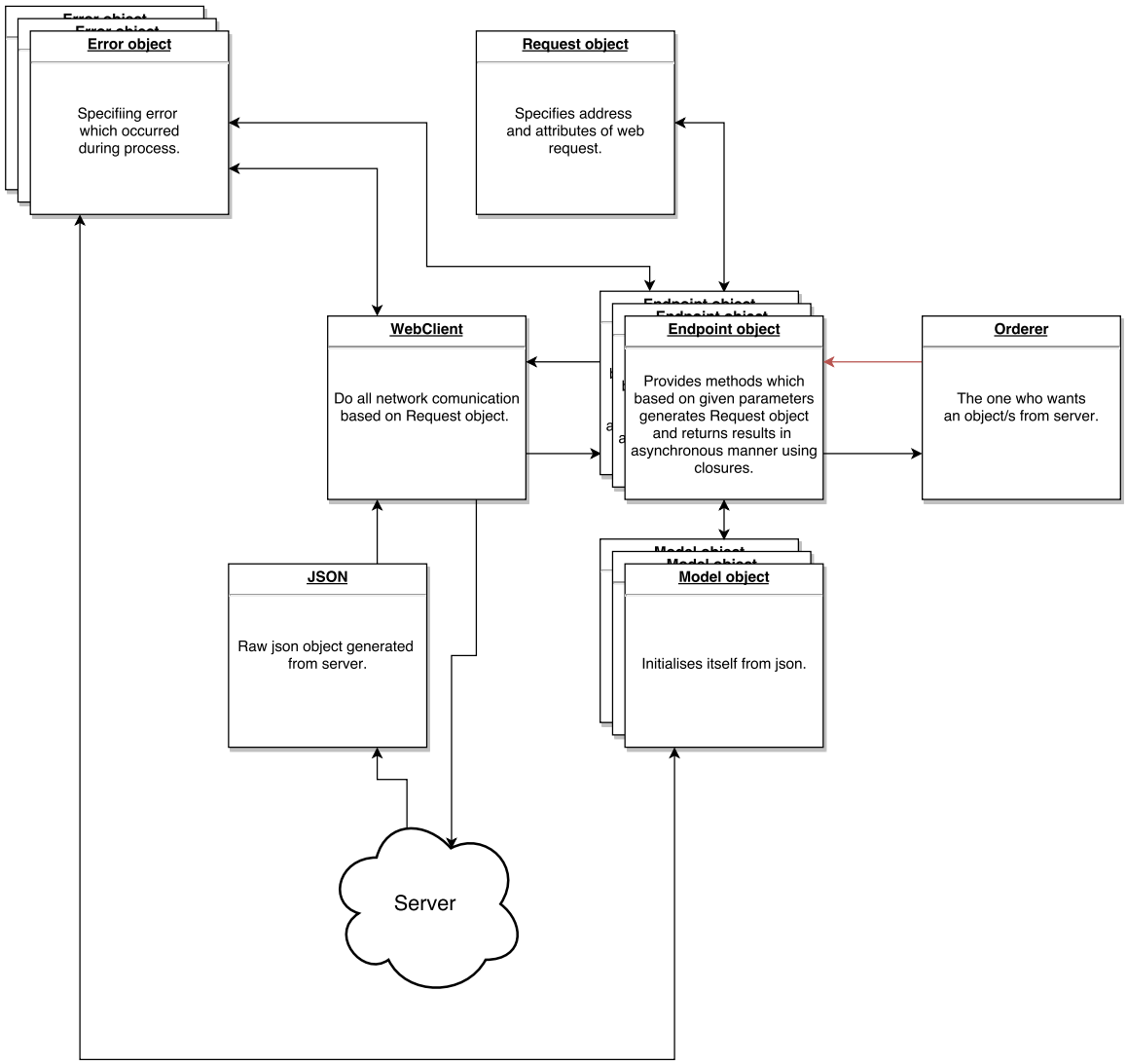
```
Alamofire.request("https://httpbin.org/get").responseJSON { response in
    print(response.request) // original URL request
    print(response.response) // HTTP URL response
    print(response.data)   // server data
    print(response.result) // result of response serialization

    if let JSON = response.result.value {
        print("JSON: \(JSON)")
    }
```

Výpis 2: Ukázka práce s Alamofire frameworkem.

4.4.1 Struktura použití Alamofire ve Sportuj v Ostravě

V našem projektu existuje jeden centrální objekt starající se o veškerou komunikaci se serverem, je jediný, kdo používá Alamofire framework (viz obr. 5). Na následujícím diagramu, je orderer označení libovolného objektu, který chce získat objekt, nebo seznam objektů, z API skrze webové rozhraní. Vytváří tedy instanci korespondujícího endpointu (jeden endpoint vždy zprostředkovává obsluhu spojenou s jednou třídou objektů), ten vytváří instanci requestu, objektu, který drží informace o adrese, parametrech, typu http dotazu apod. a předává ji WebClientovy, ten na jeho základě provede dotaz na http server a obdrží json. WebClient během této operace může, ale nemusí, vytvořit několik chybových zpráv (Error object) např. pokud je server nedostupný, nebo pokud vrátí jinou odpověď než 200 OK. Endpoint poté zpět získává buď error object, nebo surový json. Nyní se pokusí vytvořit kýžený objekt, který spravuje, objekt může opět vyhodit error že se mu nepodařilo vytvořit se s dodaného jsonu, enpoint nakonec předá zadavateli buď zhotovený objekt, nebo první error, který se po cestě vyskytl.



Obrázek 5: Znázornění fungování Alamofire ve Sportuj v Ostravě.

4.5 Fabric a Crashlytics

Vzhledem k poměrně složitému způsobu distribuce aplikací na iOS. Byla pro beta testování zvolena služba Beta od Fabric. Ta na sebe balí další užitečné funkce jako framework Crashlytics, který slouží ke sbírání údajů o využívání aplikace a hlavně poskytuje zpětnou vazbu při jejím pádu na zařízení uživatele [10].

4.5.1 Instalace

Instalace frameworku probíhá pomocí CocoaPods. Následně je ještě potřeba nastavit Xcode, aby po sestavení aplikace spustila příslušný script, který informuje server Fabric o případných změnách. A v rámci AppDelegate aplikace provést inicializaci samotné knihovny.

4.5.2 Fabric Beta

Je služba sloužící k usnadnění distribuce beta verzí aplikace mezi testery. Po nainstalování do cílového projektu je po archivaci projektu spuštěn script, který upozorní aplikaci Beta o nové verzi. Ta pak automaticky nabídne upload na svůj server s možností poslat pozvánky novým, nebo informovat stávají testery o aktualizaci. V pozvánce/upozornění poté bude link, který buď dovolí uživateli aplikaci přímo stáhnout, nebo, v případě že UDID jeho zařízení není registrováno v rámci testovacích zařízení vývojáře, automaticky získá potřebné informace o zařízení a odešle je zpět vývojáři, který jej může přidat k testovaným zařízením a vytvořit nový archív.

4.5.3 Crashlytics

Při hledání a odstraňování chyb v kódu u uživatele je nejdůležitější posbírat co nejvíce informací o případné chybě. Knihovny, které se o takovou funkčnost starají mají v zásadě dvě části. Samotný framework v zařízení a serverovou část sbírající data od všech zařízení. Crashlytics je kombinací obou a funguje tak jako komplexní služba na detekci bugů.

Dále také provádí takzvanou symbolikaci. Jedná se o klíčový proces při tvorbě zprávy o chybě, při kterém je systémový Crash report soubor vygenerovaný při pádu libovolné aplikace a který obsahuje surové informace jako hexadecimální adresy paměťových buňek, čísla procesů a podobné, pro člověka těžko čitelné, informace převeden za pomocí dSYM souboru, což je soubor vnikající při kompilaci kódu, do podoby stack trace, jakou zobrazí debugger Xcode.

4.6 Služby Keychain

Keychain je systémová klíčenka dostupná napříč všemi operačními systémy Apple, slouží k bezpečnému uchováváni hesel, klíčů, certifikátů, poznámek apod. v zašifrované podobě. Jednou z výhod keychain je také provázanost s iCloud a možnost nechat si zabezpečená data synchronizovat mezi zařízeními. Úroveň zabezpečení je možné nastavit pro každý záznam klíčenky na jednu ze 4 možností:

Always Záznam je dostupný vždy, bez ohledu na stav zařízení (zamčeno / odemčeno).

After First Unlock Záznam je zpřístupněn po prvním odemčení zařízení po restartu.

When Unlocked Záznam je dostupný pouze je-li zařízení odemčeno.

When Passcode Set Obdobně jako When Unlocked, ale vyžaduje aby bylo heslo nastaveno.

Klíčenka také dovoluje pro přístup místo klasického hesla využití senzoru otisků prstů Touch ID.

4.6.1 KeychainAccess framework

Systémová API pro používání keychain je stále dostupné pouze v Objective-C, pro jeho využití je tedy nutné použít bridging header a nejlépe také nějaký wrapper, který zabalí základní funkce pro ukládání a zpětné získávání objektů, které bohužel není tak jednoduché jako například práce s UserDefaults, nezabezpečenou alternativou klíčenky.

Naštěstí existuje spousta frameworků, které zaštiťují práci wrapperu a zprostředkovávají keychain v pro vývojáře mnohem příjemnější formě. Ve Sportuj v Ostravě byl použit KeychainAccess. Jedná se o důvěryhodný projekt, který má na GitHub téměř 2700 hvězdiček. Jeho instalace probíhá obdobně jako Alamofire tedy pouhým přidáním do podfile. Po instalaci je jeho použití jednoduché a značně připomíná práci s UserDefaults, ale nabízí i pokročilé možnosti konfigurace, jako synchronizaci s iCloud, použití Touch ID, nastavení úrovní zabezpečení atp.

```
// Storing password to keychain
let keychain = Keychain(service: "com.example.company")
keychain["key"] = "01234567-89ab-cdef-0123-456789abcdef"

// Retrieving password from keychain
if let password = keychain["key"] {
// Password exists here
} else {
// No password existed at a time
}
```

Výpis 3: Ukázka práce s KeychainAccess frameworkem.

4.7 Lokalizace

Aplikace by měla podle zadání být multijazyčná. Xcode na takovou možnost pamatuje a nabízí vestavěný nástroj, kdy veškerý text, který se vypisuje do UI zabalíme jako NSLocalizedString. Xcode jej pak překládá na základě dodaného souboru Localizable.strings.

iOS nabízí vývojářům přes 100 možných jazykových mutací dle země a regionu. A také možnost využít exportu všech řetězců, které se nalézají ve StoryBoardech do .xliff souboru. Což je standardizovaný formát založený na XML určený právě pro překlady textů. Ten je po přeložení možné naimportovat zpět do Xcode, které poté použije správný jazyk na základě jazyka zařízení.

4.7.1 Localize-Swift

V aplikaci jsme se nicméně rozhodli pro framework Localize-Swift který nabízí další výhody při multijazyčných aplikacích. Jednak to je mnohem pohodlnější syntax kdy na vybraný string stačí zavolat metodu .localized() dále umožňuje současně využívat lokalizační soubory které Xcode vygeneroval z UI. Přidává vlastní script, který v kódu vyhledá všechny výskyty .localized() a připravý pro ně šablonu pro překlad a nakonec umožní uživateli přepínat mezi různými jazykovými verzemi aplikace bez nutnosti změny jazyka zařízení.

```
// Globally changing app's language
Localize.setCurrentLanguage("cs")
// Localizing string in code
textLabel.text = "Hello World".localized()
```

Výpis 4: Ukázka práce s Localize-Swift frameworkem.

4.8 Facebook SDK

K přihlášení uživatelů do aplikace má být k dispozici krom klasického způsobu pomocí emailu a hesla také možnost přihlásit se skrze Facebook. Ten nabízí SDK nejen pro příhlášení, ale i pro analýzu používání aplikace, komplexní statistiky o uživatelích apod. Od července je vývojářská sada dostupná, sic zatím stále jako betaverze, i v programovacím jazyce Swift.

4.8.1 Instalace

Instalace probíhá v několika fázích, nejprve je třeba na Facebooku zaregistrovat mobilní aplikaci, která se skrze něj bude přihlašovat, je proto nutné zadat její název a hlavně App ID. Poté je vygenerováno unikátní číslo aplikace, které se použije později v projektu.

Přidání Facebook SDK do projektu probíhá standardně pomocí CocoaPods. Dále je třeba upravit soubor Info.plist, souboru, který se nachází v každém Xcode projektu a ve kterém jsou uloženy všechny základní konfigurační informace aplikace. Konkrétně je třeba přidat URL schéma na které bude aplikace odpovídat, to sestává z předpony fb + ID vygenerované Facebookem. Samotné přihlášení probíhá dle protokolu OAuth2, po kliknutí na přihlášení je uživateli otevřeno okno prohlížeče kde se přihlásí a je přesměrován právě přes patřičné URL schéma zpět do aplikace. Ta ve svém delegátu přesměruje příchozí komunikaci z Facebooku do jeho SDK. Dále se do Info.plist přidá ještě samotné ID a jméno aplikace, které byli vytvořeny při úvodní registraci.

4.8.2 Použití

Facebook SDK definuje také několik UI prvků, které kopírují defaultní vzhled sociální sítě a dají se použít mít systémových. Například v případě přihlášení je to LoginButton, který po kliknutí sám spustí přihlašovací proces, krom toho nabízí protokol delegáta, kterého informuje o probíhajících akcích a jejich výsledku. K získání informací o přihlášeném uživateli slouží volitelná struktura AccessToken dostupná jako statická položka current dané struktury.

```
// Adding default Facebook button to view with specific permissions.
let loginButton = LoginButton(readPermissions: [ .PublicProfile ])
loginButton.center = view.center
view.addSubview(loginButton)
```

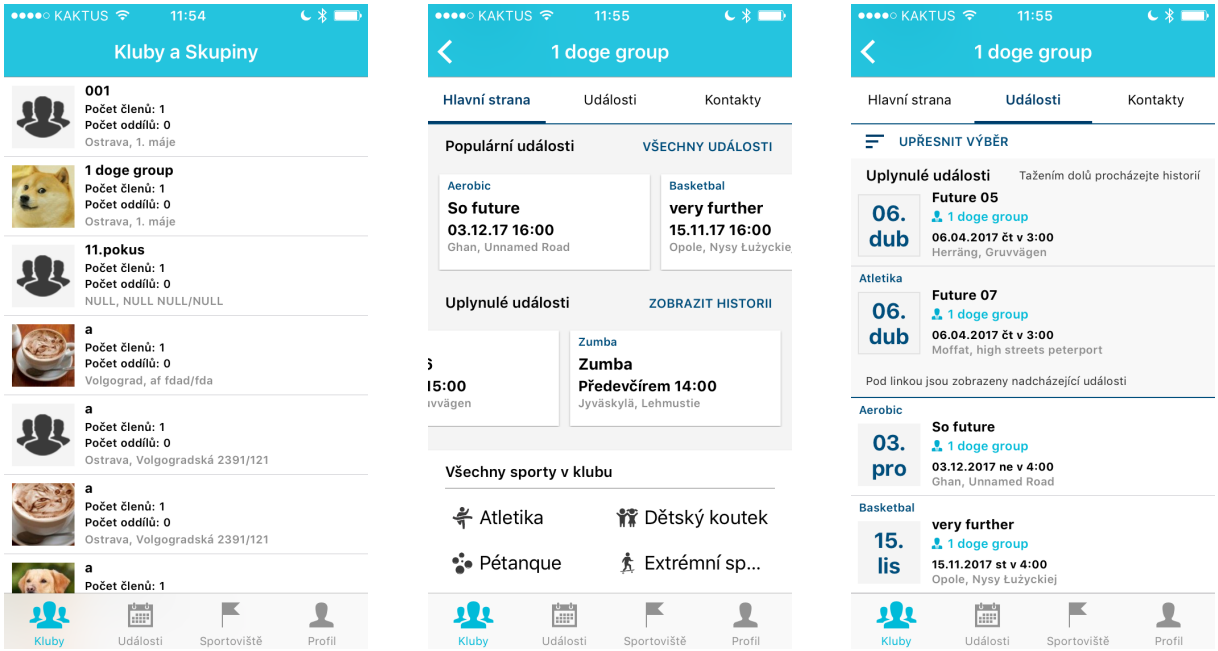
Výpis 5: Ukázka přidání výchozího přihlašovacího tlačítka.

```
if let accessToken = AccessToken.current {
   // User is logged in, use 'accessToken' here.
}
```

Výpis 6: Ukázka kontroly přihlášeného uživatele.

4.9 Aktuální stav k 9. 4. 2017

Aplikace se aktuálně skládá z 10 tříd typu UIViewController, obsahuje 4 třídy Endpoint, které zprostředkovávají 8 typů modelových objektů, 3 třídy zajišťující komunikaci s API a následné zpracování JSONu, 1 třídu zpravující přihlášení uživatele, extensions nad 9 třídami, 2 znovupoužitelná view, a 8 různých typů buňek pro UICollectionView, nebo UITableView. A strávil jsem nad ní již přes 150 hodin.



Obrázek 6: Snímky obrazovky z aplikace k 9. 4. 2017.

5 Závěr

Má praxe ve firmě probíhala, dalo by se říci, ve dvou fázích, ze začátku jsem byl zásobován materiály, ať už webovými tutoriály, nebo staršími původními zdrojovými kódy, ze kterých jsem se měl učit, udělat nějaké drobné úpravy, popřípadě konzultovat čemu nerozumím.

V druhé polovině jsme začali s vývojem skutečné aplikace, Sportuj v Ostravě. Vývoj probíhal v týdenních cyklech, vždy jsme se domluvili, jak bych měl dále pokračovat, co a jakým způsobem napsat a po dokončení jsem dostal na svoji práci review od programátora, který na mě dohlížel. Zapracoval jsem případné úpravy a pokračoval dále. Nedá se tak říci, že bych na aplikaci pracoval sám, ale prošel mi tak pod rukama každý řádek kódu, který se v aplikaci vyskytuje.

Na práci na Sportuj v Ostravě se budu dále podílet a aplikace by měla do začátku léta dorazit do AppStore.

Hodnocení svého kódu jsem dostával nejčastěji v psané formě, dostal jsem seznam věcí, které se dali, nebo měli udělat jinak. Mou nejčastější chybou bylo zbytečné přidělávání si práce, často jsem tak objevoval různé zkratky, které Swift nabízí během vývoje a můj kód tak byl delší, než by nezbytně musel. Jednalo se tak především o syntaktické chyby než o logické a většina mého kódu tak v aplikaci zůstala.

Jelikož pro mě byli Swift a Cocoa Touch na konci léta naprosto nové a neměl jsem do té doby s nimi žádné zkušenosti, musel jsem se spoustu věcí naučit sám. Některé obecné postupy v oblasti programování, které jsem si odnesl ze školy, jsou ale univerzální a určitě se tak přenesli i do mé praxe. Pro mě nejdůležitější předměty tak byli Programování II a Vývoj informačních systémů, ze kterých jsem si odnesl principy objektového programování a využívání návrhových vzorů.

Pro mě osobně byla celá zkušenost s praxí rozhodně zajímavá, jsem rád, že jsem si vybral tento způsob vypracování bakalářské práce a doufám, že se práci s iOS a Swift budu moci věnovat i do budoucna.

6 Zdroje

[1] AstrumQ [online]. [cit. 2017-04-01]. Dostupné z: `https://astrumq.com/` Oficiální stránky firmy AstrumQ.

[2] IDC [online]. [cit. 2017-03-24]. Dostupné z: `http://www.idc.com/` Americká analytická a poradenská firma se specializací na IT, telekomunikace a vývoj software.

[3] Apple [online]. [cit. 2017-03-24]. Dostupné z: `https://apple.com/` Oficiální stránky firmy Apple.

[4] Apple Developer [online]. [cit. 2017-03-24]. Dostupné z: `https://developer.apple.com/` Oficiální stránky určené pro vývojáře pro kteroukoli platformu Apple. Dokumentace, best practices, UX guidelines.

[5] Android Developers [online]. [cit. 2017-03-24]. Dostupné z: `https://developer.android.com/` Oficiální stránky pro vývojáře pro OS Android.

[6] TIOBE [online]. [cit. 2017-03-24]. Dostupné z: `http://tiobe.com/` Nizozemská společnost pravidelně vydávající žebříček popularity programovacích jazyků.

[7] Swift [online]. [cit. 2017-03-24]. Dostupné z: `https://swift.org/` Oficiální stránky open source projektu Swift.

[8] Manbolo Blog [online]. [cit. 2017-03-24]. Dostupné z: `http://blog.manbolo.com/` Blog vývojářského týmu stojícího za několika herními tituly pro iOS i OS Android.

[9] Alamofire [online]. [cit. 2017-03-24]. Dostupné z: `https://www.github.com/Alamofire/Alamofire/` Stránky projektu Alamofire na serveru github.com.

[10] Fabric [online]. [cit. 2017-03-24]. Dostupné z: `https://fabric.io/` Webová prezentace a dokumentace k frameworkům společnosti Fabric.

[11] KeychainAccess [online]. [cit. 2017-03-24]. Dostupné z: `github.com/kishikawakatsumi/KeychainAccess/` Stránky projektu KeychainAccess na serveru github.com

[12] Localize-Swift [online]. [cit. 2017-03-24]. Dostupné z: `https://github.com/marmelroy/Localize-Swift/` Stránky projektu Localize-Swift na serveru github.com

[13] Facebook for Developers [online]. [cit. 2017-03-24]. Dostupné z: `https://developers.facebook.com/` Prezentace dokumentace Facebook SDK.